APMA920 Project
Developing a method of computing the boundary of the polynomial numerical hull

Colin Macdonald

2002/12/16

The Problem With Eigenvalues

- The behavior of many problems in applied math is analyzed by considering the eigenvalues.

- This can be a mistake if the problem involves non-normal matrices.

> Eigenvalues *do not* tell the whole story if a matrix is non-normal.

- What alternatives do we have?

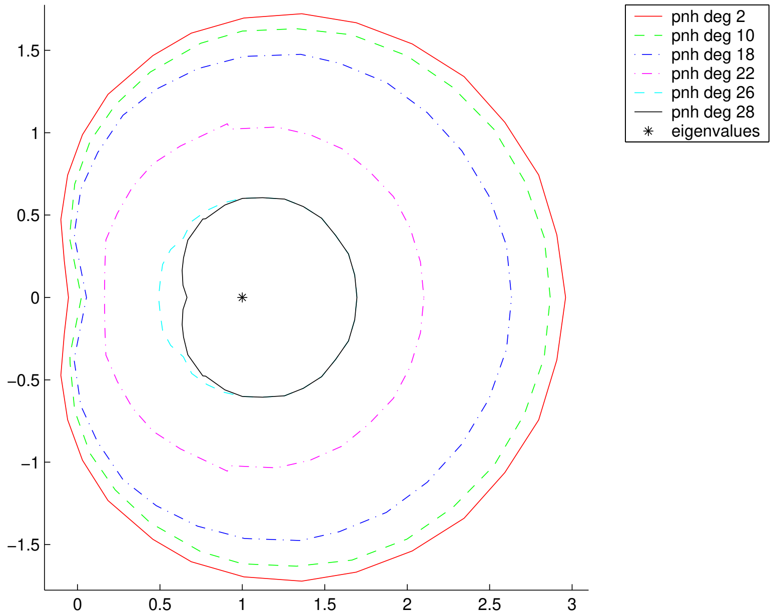
  ε-pseudospectra, *polynomial numerical hulls*

The Polynomial Numerical Hull of Degree k

$$\mathcal{G}_k(A) = \{z \in \mathbb{C} : ||p(A)|| \geq |p(z)| \forall p \in \mathcal{P}_k\}$$

The PNH tells us the behavior of polynomials of degree k or less applied to the matrix.

$$A = \begin{pmatrix} 1 & 1 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix}$$



PNH of A of various degrees.

An Example

- Consider *Gauss-Seidel* for solving a linear system $Cx = b$:

$$\boxed{x_{k+1} := Ax_k + c} \tag{1}$$

with $A = I - L^{-1}C$ and $c = L^{-1}b$ where L is the lower triangle of C.

- Eigenvalue analysis of A^k states

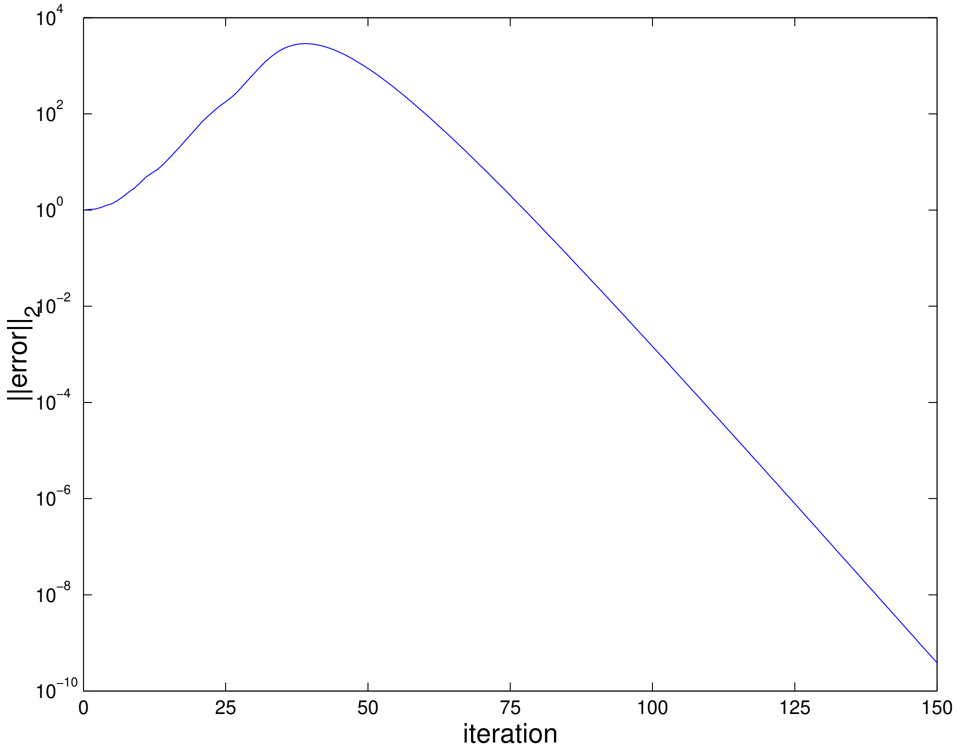
$$\boxed{\text{Iteration (1) converges as } k \to \infty \text{ iff } \rho(A) < 1.}$$

- What happens for *small k*? Is the error sequence monotonically decreasing?

Gauss-Seidel Example

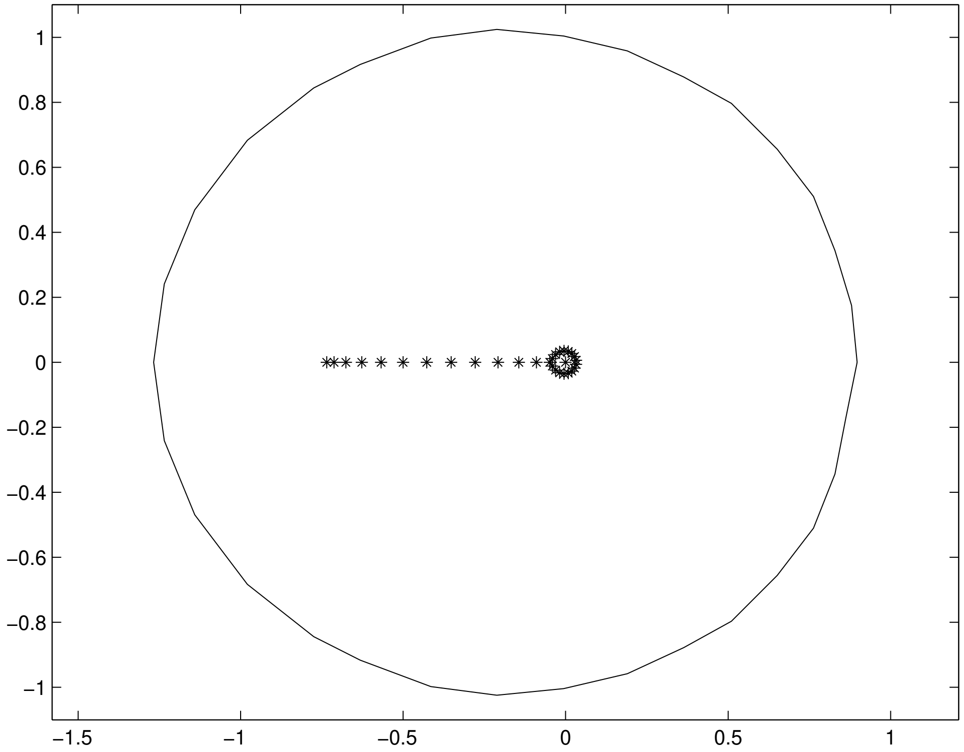
$$C = \begin{pmatrix} 1 & -1.16 & & \\ .16 & 1 & \ddots & \\ & \ddots & \ddots & -1.16 \\ & & .16 & 1 \end{pmatrix}$$

$$A = I - L^{-1}C$$



Eigenvalue analysis tell us that $\rho(A) \approx .73$ and Gauss-Seidel should converge to the solution. However it fails to mention the *initial large growth in error*.

Polynomial Numerical Hull for Gauss-Seidel



The PNH of degree 29

For $k \leq 29$, $||A^k||_2$ behaves like $|z^k|$ for $z \in \mathcal{G}_k(A)$. The largest modulus of a point in $\mathcal{G}_{29}(A)$ is 1.26. Error must *grow to at least* $1.26^{29} \approx 800$ *after* 29 *iterations* of Gauss-Seidel.

Computing The Polynomial Numerical Hull

- Recall:
$$\mathcal{G}_k(A) = \{z \in \mathbb{C} : ||p(A)|| \geq |p(z)| \forall p \in \mathcal{P}_k\}$$

- An equivalent form:
$$\mathcal{G}_k(A) = \left\{ z \in \mathbb{C} : \min_{p \in \mathcal{P}_k(0)} ||p(B = A - zI)|| \geq 1 \right\}$$
where $\mathcal{P}_k(0)$ is the set of polynomials with $p(0) = 1$ of degree $\leq k$.

- Can *check if a point z is in $\mathcal{G}_k$* by computing $\min_{p \in \mathcal{P}_k(0)} ||p(B)||$ and comparing the result to 1.

- SDPT3 [4], a matlab library for semidefinite programming, will perform this calculation. We use this as a black box code.

Detecting the Boundary of The Polynomial Numerical Hull

- Easy to form a grid and test every point to see if its in $\mathcal{G}_k$ — *This is too slow!*

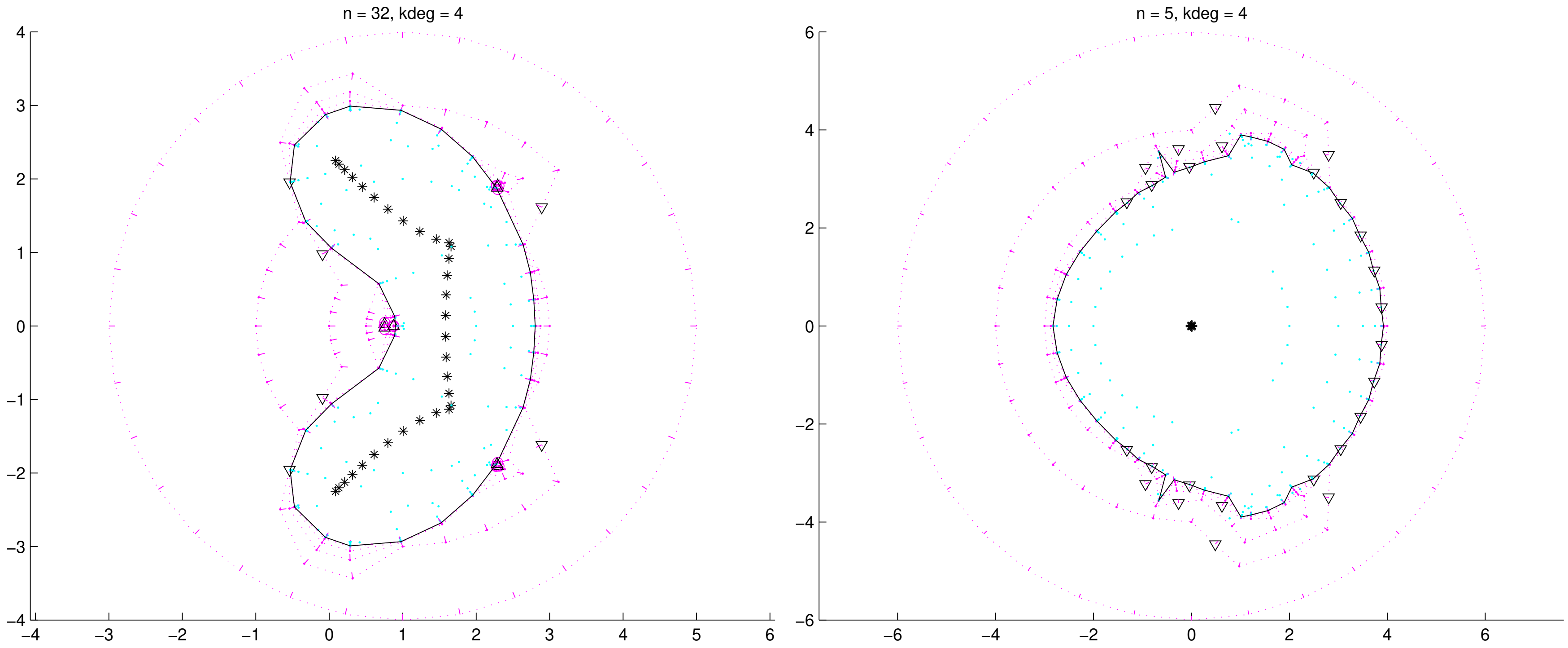
- We develop an algorithm which detects the boundary of $\mathcal{G}_k$ by surrounding it with a outer estimate and then contracting that estimate inwards. The algorithm has the following features:
  - Adds/removes nodes as necessary to produce a smooth estimate of the boundary.
  - $\mathcal{G}_{k+1}$ is nested inside $\mathcal{G}_k$ so the output of the computation for a lower degree PNH can be used as the initial guess for higher degrees.

The Boundary Finding Algorithm

- Start with `bdry` (an outer estimate of the boundary), `steps` (initial stepsizes for each node in `bdry`), `converged` (a boolean vector specifying whether each node satisfies a given tolerance `TOL`).

- while $\sum$ `converged` $\neq 0$ do
  - For each node `bdry(i)` with `converged(i)=0` do
    - estimate the inward normal
    - beginning with a stepsize `steps(i)`, perform several steps of a bisection search in the direction of the inward normal to. Store the final stepsize in `steps(i)`. Check if the node has converged to within `TOL` of the boundary and set `converged(i)` accordingly.
  - Check for nodes which are to close/to far from their neighbours and add/remove nodes as necessary.

Some Results



Acknowledgments

Thanks to Dr. Anne Greenbaum for supplying some very useful source code on short notice and for directing me to her paper [2].

References

[1] R. Burden, J. Faires, *Numerical Analysis*, 7th edition, Wadsworth Group, (2001).

[2] A. Greenbaum, *Generalizations of the field of values useful in the study of polynomial functions of a matrix*, Lin. Alg. Appl. 347 (2002), 233–249.

[3] A. Greenbaum, *Theory and applications of the polynomial numerical hull*, outline of presentation, private communication, (2002).

[4] K. Toh, M. Todd, R. Tutuncu, *SDPT3—a Matlab software package for semidefinite programming*, Optim. Meth. Softw. (1999).